GCP OptIC Project

Details on the submission by Stephen Roxburgh

Method overview:
The estimation method I use is actually a hybrid of two methods - a genetic algorithm (GA) followed by downhill simplex. I have developed this method for estimating model parameters for terrestrial carbon accounting (e.g. Roxburgh *et al.* 2006), among other purposes. It has been my experience that the GA is good at scanning the available parameter space to identify the region of the 'true' or 'global' optimum (and thus avoiding the premature convergence in complex terrain that can occur with gradient methods), but is slow to converge on a satisfactorily precise solution. I therefore run the GA first to find a solution that is (hopefully) in close proximity to the global minimum, and then send that solution to the downhill simplex for final 'fine-tuning'.

Algorithms:
I use a simple continuous implementation of the GA taken from Haupt & Haupt (1998), that uses cost-weighting selection of parents for mating, and imposes elitism (where the best solution is retained through subsequent generations). The downhill simplex algorithm ('Amoeba') is taken from Press *et al*. (1986). Numerical integration of the model is performed using the algorithm 'Odeint', also from Press *et al*. (1986). The method is coded within a Borland Delphi application, custom-written for this project (see appendix). All calculations are done using Delphi's native 10-byte 'extended' data type, with 19-20 significant digits.

Method implementation
(a) Fitted parameters
Six parameters were fitted; the four model parameters (p_1, p_2, k_1, k_2), plus the two initial conditions (x_1, x_2).

(b) Cost function:
For this exercise I used two different cost-functions, to explore the sensitivity of the solutions to the cost-function specifications.

The first was the unweighted SS:

$$\sum_{t=1}^{10000}\left(z_{1,t,Obs}-z_{1,t,Mod}\right)^2+\sum_{t=1}^{10000}\left(z_{2,t,Obs}-z_{2,t,Mod}\right)^2$$

The second weighted each SS by the grand mean of each *z*:

$$\frac{\sum_{t=1}^{10000}\left(z_{1,t,Obs}-z_{1,t,Mod}\right)^2}{\bar{z}_1}+\frac{\sum_{t=1}^{10000}\left(z_{2,t,Obs}-z_{2,t,Mod}\right)^2}{\bar{z}_2}$$

I also tried weighting each deviation by the observed value, i.e.

$$\sum_{t=1}^{10000}\frac{\left(z_{1,t,Obs}-z_{1,t,Mod}\right)^2}{z_{1,t,Obs}}+\sum_{t=1}^{10000}\frac{\left(z_{2,t,Obs}-z_{2,t,Mod}\right)^2}{z_{2,t,Obs}}$$

but a bit of trial and error found it did not produce fits as good as the first two, so for the full runs I just used the first two.

Postscript
In response to request from Cathy Trudinger, a third cost function was run, that weighted the SS of each x_i by the SD^2 (i.e. the variance) of the residuals for each dataset (that were provided):

```
Experiments T1, T4 and T6: 0.3, 1.5 (for x1 and x2, respectively)
Experiment  T2: 0.5, 1.5
Experiment  T3: 0.35, 0.6
Experiment  T5: 0.35, 1.2
Experiments A, C and D: 0.8, 4.0
Experiment  B: 3.0, 12.0
Experiment  E: 0.4, 0.9
Experiment  F: 0.07, 0.8
Experiment  G: 1.0, 2.6
Experiment  H: 0.7, 3.5
Experiment  I: 5.0, 5.0
Experiment  J: 0.7, 2.5
```

Postscript #2
On 17 July 2006 an additional 5 training datasets were received (T7-T11). These are incorporated into the results tables below, and the residual SD's are:

```
Experiment T7: 0.8, 2.2 (for x1 and x2, respectively)
Experiments T8, T9, T11: 0.3, 1.5
Experiment T10: 0.9, 3.0
```

GA:
An initial population of 600 solutions was generated at random within the allowed parameter space (as defined by the constraints), and the cost function of each was calculated. This was to give an initial 'shotgun' view of the parameter space. The best 48 solutions were then retained, and allowed to evolve for a further 250 generations (with a mutation probability of 0.03). The best solution after 250 generations was then passed onto the simplex. Note that a more flexible stopping rule could have been adopted (e.g. keep evolving until no gain occurs after *x* generations), however through a little trial and error it seemed that a constant evolution time seemed to work OK, so I kept it simple (I tried a subset of runs with a cut-off 2000 generations, and the results were the same). Also, the downhill simplex has a flexible cutoff (see below).

Downhill simplex:
The best solution from the GA was then fed to the simplex, that then refined it until there was no further improvement of the cost function (controlled by the 'Amoeba' parameter *ftol*, that was set to 1e-18).

Each of the 21 datasets was therefore optimised using two different cost functions (yielding 32 result files). Each of these combinations was also replicated three times to check the stability of the method. For each dataset/cost function combination the three replicate runs converged to the same solution.

Results summary: Parameters

Cost function: unweighted

Dataset	Func. Min.	p_1	p_2	k_1	k_2	$x1_0$	$x2_0$
T1	23924.10	1.06288	1.31055	0.22947	0.07998	6.05047	12.09362
T2	23523.07	1.06128	1.32670	0.22947	0.07976	5.92720	12.15467
T3	4512.54	1.00183	1.40424	0.22973	0.08051	6.14585	11.61704
T4	23939.32	1.06030	1.32819	0.22967	0.07976	6.29318	11.74616
T5	13089.37	1.01500	1.40759	0.23211	0.07844	1.19526	9.36268
T6	1170.10	1.04950	1.34535	0.23008	0.07899	6.08333	13.05788
T7	54160.76	1.14804	1.20464	0.22502	0.07824	5.73414	12.72616
T8	28764.02	0.60726	1.91955	0.23957	0.08481	7.02934	11.93941
T9	178.99	0.92990	1.78075	0.22698	0.07741	0.44360	0.19660
T10	99076.31	1.09258	1.21348	0.23010	0.08021	6.38371	10.99954
T11	93724.94	1.06848	0.50000	0.24823	0.08901	5.86626	12.30859
A	170403.88	2.53730	2.22170	0.11007	0.03094	4.30607	26.96224
B	1547536.65	2.37232	4.39634	0.10986	0.01098	92.35268	908.14085
C	170064.35	2.56052	2.21772	0.10974	0.03091	1.43035	6.38316
D	170105.13	2.45645	2.42764	0.10978	0.03102	5.46663	23.06772
E	7848.60	1.12640	1.56145	0.23072	0.11024	0.15850	0.05958
F	6404.38	4.61891	1.44975	0.62731	0.01101	0.21421	6.83502
G	76183.51	2.36554	2.54357	0.11051	0.03111	5.81981	25.52992
H	109742.21	2.07812	3.14642	0.11072	0.03118	13.62959	21.19807
I	503406.63	0.85319	2.43926	0.03307	0.02504	53.89397	81.16828
J	53515.18	2.37719	2.56752	0.11002	0.03110	3.56917	18.68199

Cost function: grand mean weighted

Dataset	Func. Min.	P_1	p_2	k_1	k_2	$x1_0$	$x2_0$
T1	2746.96	1.05157	1.33038	0.22980	0.08001	6.08063	12.05663
T2	2978.54	1.04434	1.35559	0.22979	0.07983	5.95271	12.12424
T3	688.77	1.00125	1.43383	0.22920	0.08031	5.72240	11.98472
T4	2747.05	1.05520	1.33455	0.22971	0.07981	6.30118	11.73912
T5	1611.68	1.05358	1.35588	0.23118	0.07817	4.04129	6.40832
T6	306.21	1.04250	1.36586	0.22998	0.07896	6.08090	13.01271
T7	6852.68	1.11574	1.25923	0.22548	0.07837	5.65381	12.78319
T8	3413.32	0.79133	1.46162	0.24052	0.08446	6.73858	12.19883
T9	20.26	1.00010	1.62396	0.22710	0.07722	0.41504	0.19695
T10	12052.68	1.05903	1.26759	0.23035	0.08040	6.28985	11.10415
T11	13329.42	1.06707	0.50000	0.25142	0.08871	6.07061	12.04199
A	7209.89	2.52624	2.24795	0.10998	0.03096	4.47959	26.74928
B	2603.98	2.38949	4.46801	0.10981	0.01098	88.66598	911.61734
C	6806.75	2.49972	2.33862	0.10989	0.03096	1.68064	5.98850
D	7078.03	2.45684	2.42886	0.10986	0.03101	5.85327	22.63499
E	2251.54	1.12479	1.56330	0.23084	0.11024	0.09903	0.14856
F	840.65	4.60464	1.45033	0.62929	0.01100	0.13565	7.25569
G	3680.60	2.37777	2.51325	0.11047	0.03111	5.96376	25.38052
H	4314.20	2.15378	2.94801	0.11070	0.03115	13.39137	21.45804

I	6783.81	0.89039	2.38593	0.03307	0.02504	53.79187	81.22847
J	2445.37	2.41276	2.49460	0.10984	0.03109	3.46288	18.85145

Cost function: residual SD^2 weighted

Dataset	Func. Min.	P_1	p_2	k_1	k_2	$x1_0$	$x2_0$
T1	20329.63	1.03615	1.35915	0.23002	0.08005	6.10830	11.96787
T2	18590.21	1.02705	1.38469	0.23004	0.07991	5.96938	12.09208
T3	18057.08	1.00127	1.43467	0.22918	0.08030	5.71479	11.99123
T4	20373.34	1.04439	1.34415	0.22998	0.07992	6.31042	11.70720
T5	17482.83	1.10761	1.30442	0.22965	0.07765	5.69030	4.90413
T6	10216.29	1.03929	1.37522	0.22993	0.07895	6.09272	12.55169
T7	19192.49	1.08705	1.30891	0.22580	0.07848	5.61359	12.78677
T8	30028.06	0.97631	1.08944	0.24025	0.08375	6.47963	12.51476
T9	177.48	1.12264	1.36857	0.22680	0.07689	0.00000	1.34529
T10	1029429.31	1.10841	1.18395	0.23117	0.08004	6.56953	10.80044
T11	215756.69	1.09478	0.50000	0.25163	0.08779	6.22883	11.62094
A	20379.89	2.50697	2.29476	0.10995	0.03096	4.60254	26.48288
B	20137.71	2.39118	4.56024	0.10980	0.01098	88.39431	911.87352
C	20326.56	2.42032	2.50532	0.11002	0.03101	1.91007	5.77716
D	20326.68	2.46149	2.42907	0.10985	0.03100	6.11483	22.27543
E	15214.58	1.12412	1.56438	0.23089	0.11022	0.09737	0.15222
F	19988.89	4.60301	1.45033	0.62951	0.01100	0.11498	7.37233
G	19932.8	2.38413	2.49565	0.11047	0.03111	6.01171	25.33071
H	17348.64	2.27952	2.59140	0.11079	0.03114	13.21857	22.13616
I	20136.27	0.85319	2.43926	0.03307	0.02504	53.89397	81.16828
J	16554.86	2.44766	2.42047	0.10976	0.03107	3.40016	19.02866

NOTES:

- The estimated initial conditions for T5 were quite different to the other training sets, and visual inspection of the observed vs. expected timelines indicated that, at least for the initial timesteps, the observed values were biased downwards.
- T9 – difficulty in determining consistent initial conditions, due probably to lots of missing data for both x_1 and x_2.
- T11 – did not converge to an internal optima within the given constraints (the estimate for p_2, for all cost functions, converged to the lower bound of 0.5.

  Resetting the upper- and lower bounds of P_1 and P_2 [(0.95, 1.05) instead of (0.5, 5.0) for P_1; (1.34, 1.36) instead of (0.5, 5.0) for P_2] to give the algorithm more of a chance of finding the true parameter values of p_1=1.04 and p_2 = 1.35 did not help, as both p_1 and p_2 then got stuck on their respective lower boundaries. Nasty (I tried this only with residual SD^2 cost function weighting).

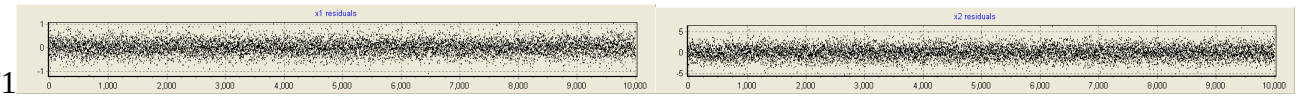
  Opening up the P_2 boundaries to (0.0, 5.0) resulted in the same pattern, with the estimate for p_2 now butting up against 0, with the new estimate of P_1 =1.67, and with a function minimum of 212565.14, which compares with 215756.69 from the standard run.

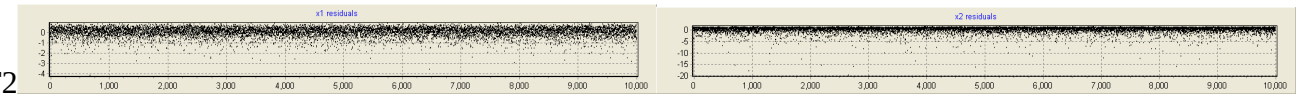
  I then went crazy, and set the bounds of p_2 to (-1000.0, 5.0). This time an internal optima was found, with the parameter estimates and function minimum given below:

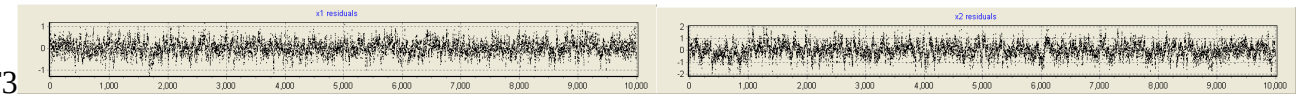
Dataset	Func. Min.	P_1	p_2	k_1	k_2	$x1_0$	$x2_0$
T11	208688.75	3.78640	-1.39083	0.20212	0.07105	5.80484	12.41601

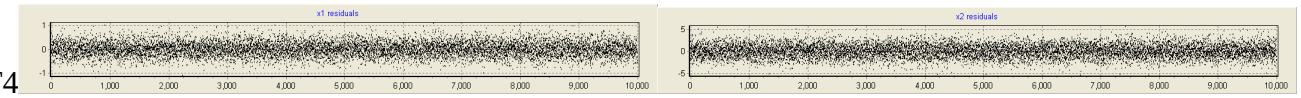
Results summary: Residuals

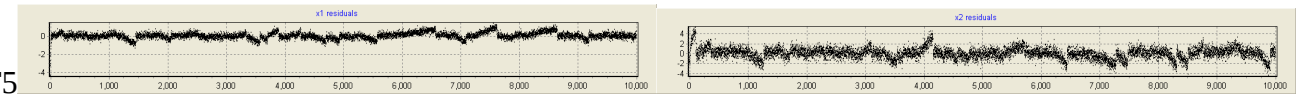
Residuals vs. time for x_1 and x_2 for the unweighted analysis (the patterns for the grand-mean weighted analysis were similar).

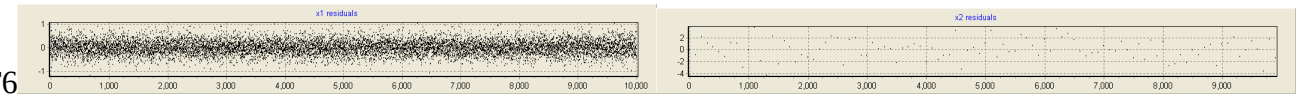
T1 

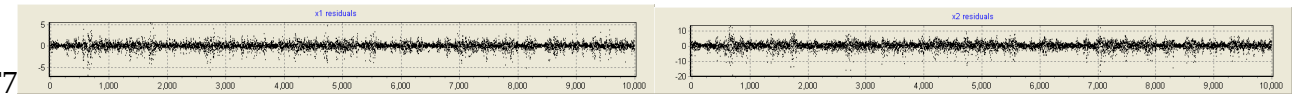
T2 

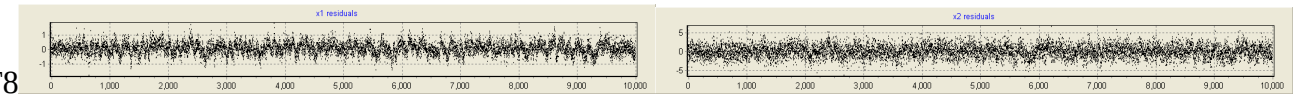
T3 

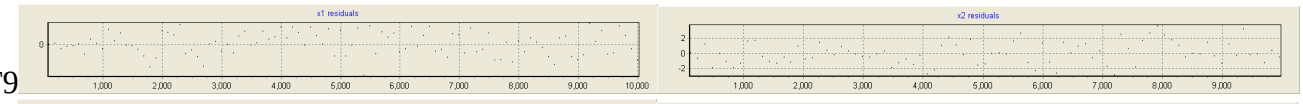
T4 

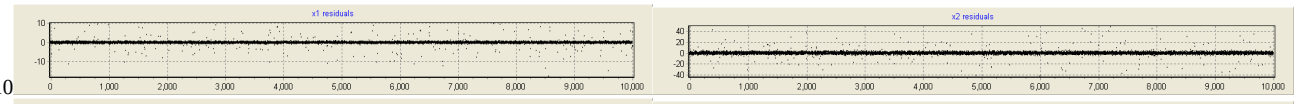
T5 

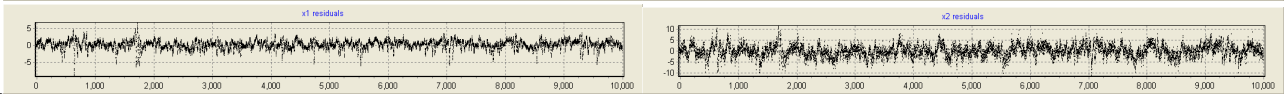
T6 

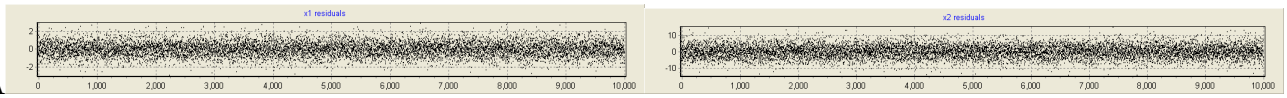
T7 

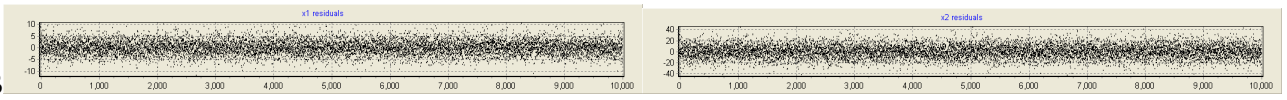
T8 

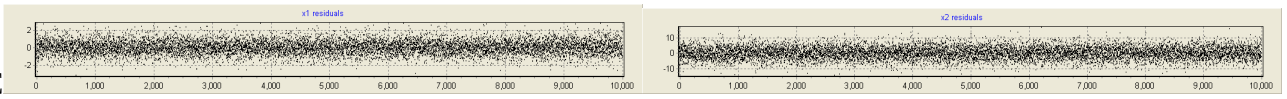
T9 

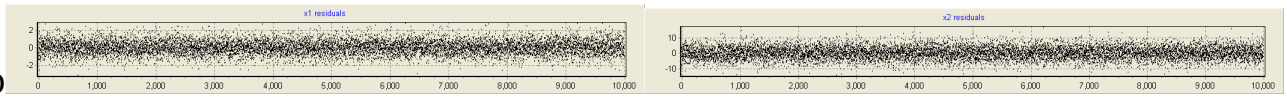
T10 

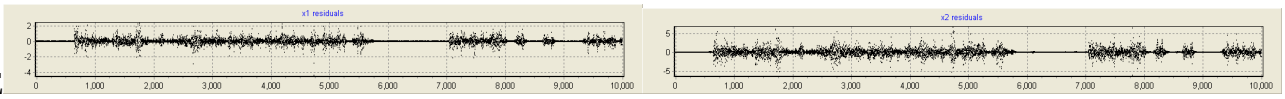
T11 

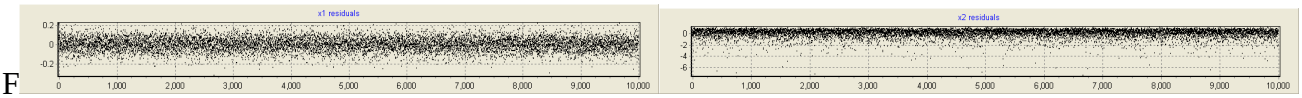
A 

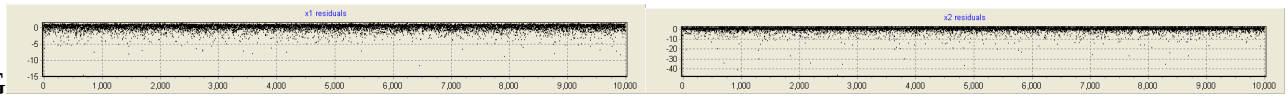
B 

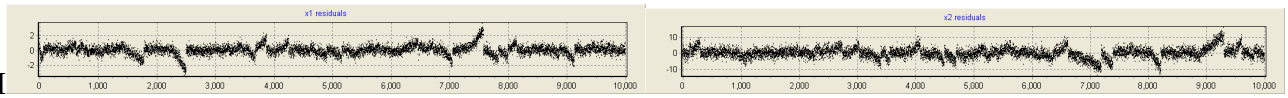
C 

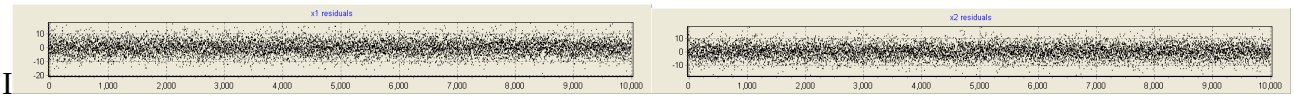
D 

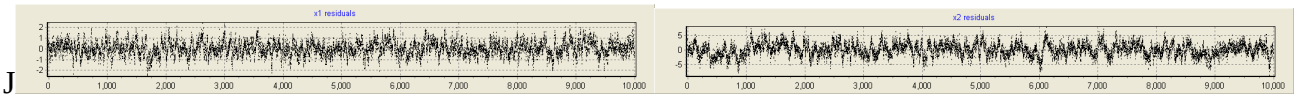
E 

F 

G 

H 

I 

J 

References

Haupt, R. & Haupt, S. 1998. *Practical genetic algorithms*. John Wiley & Sons.

Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. 1986. *Numerical recipes, the art of scientific computing*. Cambridge Univ. Press, Cambridge.

Roxburgh, S.H., Wood, S.W., Mackey, B.G., Woldendorp, G., Gibbons, P. *In press*. Assessing the carbon sequestration potential of managed forests: a case study from temperate Australia. *Journal of Applied Ecology*.

Appendix – Screen shot of the program interface

Program interface for the OptIC project. The list of numbers on the RHS are the ranked finesses of the 48 solution populations from the GA. The top two charts are the residuals. The bottom chart is the observed and modelled data, including the extension of the modelled data up to 12000 timesteps.